Subject: Re: Need IDL 8 Help
Posted by kagoldberg on Sat, 25 Jan 2014 08:55:35 GMT
View Forum Message <> Reply to Message

Here's another, related way to do this: broadcast your message to all child objects, and let the child objects decide if that message requires them to take action.

Suppose each message-able graphic object has a 'parent' keyword in the Init method where we pass the parent object's reference. Every such object then knows its parent explicitly. (You could store this information in the fields of the object, or only use it in the Init method.)

In their Init methods, objects call their parent's 'register' method, which simply stores the object reference in a big hash or list. The only argument to 'register' is the child object's reference (i.e. self, when you are within the Init method of the child).

Now, whenever the parent has ANY sort of relevant change (no matter the type) it tells every registered object, with a foreach loop, calling the 'notify' methods of those objects. The parent sends some kind of coded "type of change" or "type of message" information, plus whatever relevant arguments are necessary to interpret the change. (Lots of ways to do this, obviously.)

Then within the 'notify' methods, it's up to each child to determine if the information it is given is important to it, or if it should be ignored. I'd use a case statement in the 'notify' method to select what to do with each possible kind of notification.

You give up a small amount of efficiency by broadcasting messages to objects that may not need to know about them (since you're notifying all registered objects). BUT, you save yourself all of the trouble of having to keep track of which objects need to be the recipients of what kinds of information. That latter system could drive you mad, and I think that's the complexity that drove your question.

Unless you have a huge number of child objects, messaging all of the registered objects will happen so fast.

Another tip with object graphics and this messaging system you're creating is to delay Draw updates until everyone has been messaged, then draw all at once. If every notification prompts a re-draw, you're going to see a big slow-down.

If you code it right, you could make the notification recursive. So when an object is notified, it acts on relevant information, and then passes the notification (or a modified notification) down it ITS registered children. This makes notification distribution hierarchical, rather than flat.